Goal Change

Steven Shapiro
Computer Science Inst.
University of Leipzig
Leipzig 04109
Germany
shapiro@informatik.uni-leipig.de

Yves Lespérance
Dept. of Computer Science
York University
Toronto, ON
M3J 1P3 Canada
lesperan@cs.yorku.ca

Hector J. Levesque
Dept. of Computer Science
University of Toronto
Toronto, ON
M5S 3G4 Canada
hector@ai.toronto.edu

Abstract

Although there has been much discussion of belief change (e.g., [Gärdenfors, 1988; Spohn, 1988]), goal change has not received much attention. In this paper, we propose a method for goal change in the framework of Reiter's [2001] theory of action in the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998], and investigate its properties. We extend the framework developed in Shapiro *et al.* [1998] and Shapiro and Lespérance [2001], where goals and goal expansion were modelled, but goal contraction was not.

1 Introduction

One of the commonsense abilities people possess is the ability to predict the behaviour of others. We attribute goals and beliefs to others and assume they will act on their beliefs to achieve their goals. One strategy for endowing machines with a similar ability is to explicitly model the beliefs and goals of agents and assume they are acting to achieve their goals in accordance with their beliefs. In changing environments, the beliefs and goals of agents are likely to change as well. Although there has been much discussion of belief change (e.g., [Gärdenfors, 1988; Spohn, 1988]), goal change has not received much attention. In this paper, we propose a method for specifying and reasoning about goal change in the framework of Reiter's [2001] theory of action in the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998]. We thus inherit Reiter's solution to the frame problem, and iterated goal change is achieved seamlessly using sequences of actions. This work extends the framework developed in Shapiro *et al.* [1998] and Shapiro and Lespérance [2001], where goals and goal expansion were modelled but goal contraction was not. It has been incorporated into the Cognitive Agents Specification Language and Verification Environment [Shapiro *et al.*, 2002] for specifying and verifying multiagent systems.

We model agents that adopt goals if they are requested to do so and they do not already have a conflicting goal. They maintain their goals unless they are explicitly requested to drop them by the agent who requested the goal in the first place. We first define goals using an accessibility relation, W, which picks out the situations that the agent wants to be the case. We then give a successor state axiom for W that is affected by two communicative actions: REQUEST actions, which cause agents to adopt goals, and CANCELREQUEST actions, which cause agents to drop goals that had been adopted following prior requests. We then consider some properties of our axiomatization: expansion, contraction, and persistence. Finally, we identify the restrictions on the accessibility relations that give us positive and negative introspection of goals, and show that these restrictions persist, if they are asserted of the initial situations.

The remainder of the paper is organized as follows. In Sec. 2, we give a brief review of the situation calculus, Reiter's theory of action, and Shapiro *et al.*'s [1998] framework for representing knowledge expansion in the situation calculus. In Sec. 3, we introduce our framework for goal change, and in Sec. 4, we investigate properties of this framework. In Sec. 5, we present an example and in Sec. 6, we conclude and suggest avenues for future work.

2 Representation of Action and Knowledge

The basis of our framework for goal change is an action theory [Reiter, 2001] based on the situation calculus [McCarthy and Hayes, 1969; Levesque *et al.*, 1998]. The situation calculus is a predicate calculus language for representing dynamically changing domains. A situation represents a possible state of the domain. There is a set of initial situations corresponding to the ways the agent believes the domain might be initially. The actual initial state of the domain is represented by the distinguished initial situation constant, S_0, which may or may not be among the set of initial situations believed possible by the agent. The term $do(a, s)$ denotes the unique situation that results from the agent doing action a in situation s. Thus, the situations can be structured into a set of trees, where the root of each tree is an initial situation and the arcs are actions. The sequence of actions that produces a situation is called the *history* of the situation. $s \preceq s'$ ($s \prec s'$, resp.) means there is a (nonempty, resp.) path from situation s to situation s'. The initial situations are defined as those having an empty history:[1]

$$Init(s) \stackrel{\text{def}}{=} \neg\exists a, s'.s = do(a, s').$$

Predicates and functions whose value may change from situation to situation (and whose last argument is a situation) are called *fluents*. For instance, we might use the fluent $\textsc{InRoom}(Agt, R_1, S)$ to represent the fact that agent *Agt* is in room R_1 in situation S. The effects of actions on fluents are defined using successor state axioms [Reiter, 2001], which provide a succinct representation for both effect axioms and frame axioms [McCarthy and Hayes, 1969].

We will be quantifying over formulae, so we assume that we have an encoding of formulae as first-order terms (we can adapt De Giacomo *et al.*'s [2000] axioms for this purpose, but we omit the details here). Therefore, we will freely quantify over formulae here. To simplify notation, we ignore the details of the encoding and use formulae directly instead of the terms that represent them. We will use ψ to denote a formula that can contain the situation constants *Now* and *Then*, and ϕ to denote a formula that can contain *Now* but not *Then*. $\phi[s]$ ($\psi[s, s']$, resp.) will be used to denote the formula that results from substituting s for *Now* in ϕ (s for *Now* and s' for *Then*, resp.).

To axiomatize a dynamic domain in the situation calculus, we use Reiter's [2001] action theory, which consists of (1) successor state axioms for each fluent (including K and W introduced below); (2) unique names axioms for the actions, and domain-independent foundational axioms (we adopt the ones given by Levesque *et al.* [1998] which accommodate multiple initial situations, but we do not describe them further here); (3) initial state axioms, which describe the initial state of the domain and the initial mental states of the agents, and (4) the axioms to encode formulae as terms.

Moore [1985] defined a possible-worlds semantics for a logic of knowledge in the situation calculus by treating situations as possible worlds. Scherl and Levesque [2003] adapted this to Reiter's theory of action. Scherl and Levesque gave a successor state axiom for K that states how actions, including sensing actions, affect knowledge. Shapiro *et al.* [1998] adapted this axiom to handle multiple agents and INFORM actions, which we adopt here. $K(agt, s', s)$ will be used to denote that in situation s, agt thinks that situation s' might be the actual situation. Note that the order of the situation arguments is reversed from the convention in modal logic for accessibility relations. $\mathbf{Know}(agt, \phi, s)$ denotes that agt knows ϕ in s. $\textsc{inform}(infr, agt, \phi)$ is the action of the agent $infr$ informing another agent, agt, that ϕ holds. Here is Shapiro *et al.*'s [1998] successor state axiom for K.[2]

[1]We adopt the convention that unbound variables are universally quantified in the widest scope.

[2]Since we are using terms for formulae in our underlying representation, $\textsc{inform}(infr, agt, \phi) = \textsc{inform}(infr, agt, \phi')$, only if ϕ and ϕ' are the same term. It would not be difficult to modify our representation so that the equality holds if ϕ and ϕ' are the representations of equivalent formulae.

Axiom 2.1 ([Shapiro *et al.*, 1998])

$$\begin{aligned}&K(agt, s'', do(a, s)) \equiv\\ &\exists s'.K(agt, s', s) \wedge s'' = do(a, s') \wedge Poss(a, s') \wedge\\ &\qquad \forall infr, \phi.a = \textsc{inform}(infr, agt, \phi) \supset \phi[s']\end{aligned}$$

First note that for any action other than an INFORM action directed towards agt, the specification ensures that the only change in knowledge that occurs in moving from s to $do(a, s)$ is that it is known (by all agents) that the action a has been successfully executed. This is true because the K-alternative[3] situations to $do(a, s)$ are the situations that result from doing a in a K-alternative situation to s where a is possible to execute (denoted by $Poss(a, s')$). For the action $\textsc{inform}(infr, agt, \phi)$, the idea is that in moving from s to $do(\textsc{inform}(infr, agt, \phi), s)$, agt not only knows that the action has been executed (as above), but it also knows that ϕ holds. In this case, the K-alternative situations to $do(a, s)$ for agt are the situations that result from performing a in a K-alternative situation to s where a is possible to execute, except the ones where ϕ is false.[4]

As usual, we say that an agent knows a formula ϕ in s, if ϕ holds in all situations K-accessible from s, i.e., $\mathbf{Know}(agt, \phi, s) \stackrel{\text{def}}{=} \forall s'.K(agt, s', s) \supset \phi[s']$.

Following Shapiro *et al.* [1998], we say that an agent can only execute an inform action if it knows that the content of the action is true.

Axiom 2.2

$$Poss(\textsc{inform}(informer, agt, \phi), s) \equiv \mathbf{Know}(informer, \phi, s).$$

Since we are dealing with knowledge here, not belief, we assert that K is (initially) reflexive.[5]

Axiom 2.3

$$Init(s) \supset K(agt, s, s)$$

Following Scherl and Levesque, we also assert that initial situations can only be K-related to other initial situations. We use an initial state axiom for this purpose:

Axiom 2.4

$$K(agt, s', s) \wedge Init(s) \supset Init(s')$$

Note that this axiom and the successor state axiom for K imply that only situations with the same history can be K-related.

[3]We say that S' *is a K-alternative situation to* S or S' *is K-related to* S, if for some agent *Agt*, $K(Agt, S', S)$ holds.

[4]Note that here all agents are aware of all actions that occur, including inform actions, so we have a broadcast model of communication. In Shapiro and Lespérance [2001], encrypted speech acts were introduced, ensuring that only the intended recipient of a message could know its contents.

[5]While we deal here with knowledge and knowledge expansion, but not revision, note that a treatment of belief revision in our framework was given in [Shapiro *et al.*, 2000].

3 Goal Change

Just as an agent's beliefs can change, its goals should also be able to change. For example, if an agent's owner requests it to do something, the agent's goals should change to reflect the request. We extend the framework developed in Shapiro *et al.* [1998] and Shapiro and Lespérance [2001], where goals and goal expansion were modelled but they did not allow for goal contraction.

Following Cohen and Levesque [1990], we model goals using an accessibility relation over possible worlds (situations, in our case). The accessible worlds are the ones that are compatible with what the agent *wants* to be the case. Cohen and Levesque's accessibility relation for goals, G, was a subset of their accessibility relation for beliefs. This constraint precludes an agent wanting something that it believes is impossible (unless its goals are inconsistent). In our case, we define G in terms of a more primitive accessibility relation, which we call W and is intended to model what the agent wants independently of what it knows.[6] We can then define Cohen and Levesque's goal accessibility relation, G, to be the intersection[7] of W and K.

An agent's goals are future oriented. For example, an agent might want some property to hold eventually, i.e., the agent's goal is of the form **Eventually**(ψ), for some formula ψ. We evaluate formulae such as these with respect to a path of situations rather than a single situation, and we call such formulae *path formulae*. Cohen and Levesque used infinite time-lines to evaluate such formulae, but for simplicity, we evaluate path formulae with respect to finite paths of situations which we represent by pairs of situations, $(\mathit{Now}, \mathit{Then})$, such that $\mathit{Now} \preceq \mathit{Then}$. *Now* corresponds to the "current time" on the path of situations defined by the sequence of situations in the history of *Then*. Path formulae may contain two situation constants, *Now* and *Then*. For example, $\exists r.\textsc{InRoom}(\textsc{John}, r, \mathit{Now}) \wedge \neg \textsc{InRoom}(\textsc{John}, r, \mathit{Then})$ could be used to denote the goal that John eventually leaves the room he is in currently.

Our W relation, like our K relation, is a relation on situations. While K only relates situations with the same histories, W can relate situations with different histories. Intuitively, $W(agt, s', s)$ holds if in situation s, agt considers that in s' everything that it wants to be true is actually true. For example, if the agent wants to become a millionaire in s, then in all situations W-related to s, the agent is a millionaire, but these situations can be arbitrarily far in the future.

Recall that the situations in the W accessibility relation are the ones that the agent wants to actualize independently of what it knows. Following Cohen and Levesque, we want the goals of the agent to be compatible with what it knows. The situations that the agent wants to actualize should be on a path from a situation that the agent considers possible. Therefore, the situations that will be used to determine the goals of an agent will be the W-accessible situations that are also compatible with what the agent knows, in the sense that there is K-accessible situation in their history. We will say that s' $K_{agt,s}$*-intersects* s'' if $K(agt, s'', s)$ and $s'' \preceq s'$. We will suppress agt or s if they are understood from the context. We define the goals of agt in s to be those formulae that are true in all the situations s' that are W-accessible from s and that K-intersect some situation, s'':

$$\textbf{Goal}(agt, \psi, s) \stackrel{\text{def}}{=} \forall s', s''. W(agt, s', s) \wedge K(agt, s'', s) \wedge s'' \preceq s' \supset \psi[s'', s'].$$

Note that s'' corresponds to the "current situation" (or the current time) in the path defined by s'.

As noted in Konolige and Pollack [1993], one has to be careful when using this definition of a goal. Suppose an agent has a conjunctive goal, $\psi \wedge \psi'$. According to this definition of **Goal**, both ψ and ψ' are also goals. Therefore, we could imagine a rational agent working to achieve one of the conjuncts as a subgoal. But it is easy to think of circumstances where achieving only one component of a conjunctive goal is undesirable. For example, I may have as a goal to be holding the bomb *and* that the bomb be defused. However, I want these to be true simultaneously and I do not have the goal to be holding the bomb if it is not also defused. As Konolige and Pollack did for their intention modality (I), we consider formulae that are the agent's "only goals", i.e., formulae that are true in all and *only* the W-paths:

$$\begin{aligned}\textbf{OGoal}(agt, \psi, s) &\stackrel{\text{def}}{=} \\ &\forall s', s''. K(agt, s'', s) \wedge s'' \preceq s' \supset \\ &\qquad (W(agt, s', s) \equiv \psi[s'', s']).\end{aligned}$$

We now give the successor state axiom for W. $W^{+}(agt, a, s', s)$ ($W^{-}(agt, a, s', s)$, resp.), which is defined below, denotes the conditions under which s' is added to (dropped from, resp.) W due to action a:

Axiom 3.1

$$\begin{aligned}&W(agt, s', do(a, s)) \equiv \\ &(W^{+}(agt, a, s', s) \vee (W(agt, s', s) \wedge \neg W^{-}(agt, a, s', s))).\end{aligned}$$

An agent's goals are expanded when it is requested to do something by another agent. After the $\textsc{request}(requester, agt, \psi)$ action occurs, agt should adopt the goal that ψ, unless it currently has a conflicting goal (i.e., we assume agents are maximally cooperative). Therefore, the $\textsc{request}(requester, agt, \psi)$ action should cause agt to drop any paths in W where ψ does not hold. This action is taken into account in the definition of W^{-}:

$$\begin{aligned}&W^{-}(agt, a, s', s) \stackrel{\text{def}}{=} \\ &\exists requester, \psi, s''. a = \textsc{request}(requester, agt, \psi) \wedge \\ &\quad \neg\textbf{Goal}(agt, \neg\psi, s) \wedge K(agt, s'', s) \wedge s'' \preceq s' \wedge \\ &\quad \neg\psi[do(a, s''), s'].\end{aligned}$$

[6] Some authors contend that an agent's wants or desires should be allowed to be inconsistent. Since we model what the agent wants using a standard possible-worlds semantics, we cannot model inconsistent desires (unless the agent wants everything). We do not feel that this is a serious drawback for practical applications of our theory.

[7] As we will see below, the operation used on W and K to obtain G is not quite intersection, but a related operation.

According to this definition, s' will be dropped from W, if for some *requester* and ψ, a is the $\text{REQUEST}(requester, agt, \psi)$ action and agt does not have a conflicting goal, i.e., the goal that $\neg\psi$ in s[8], and s' K-intersects some s'' such that ψ does not hold on the path $(do(a, s''), s')$. The reason that we check whether $\neg\psi$ holds at $(do(a, s''), s')$ rather than at (s'', s') is to handle goals that are relative to the current time. If, for example, ψ states that the very next action should be to get some coffee, then we need to check whether the next action after the request is getting the coffee. If we checked $\neg\psi$ at (s'', s'), then the next action would instead be the REQUEST action.

If the agent gets a request for ψ and it already has the goal that $\neg\psi$ then it does not adopt the goal that ψ, otherwise its goal state would become inconsistent and it would want everything. This is a simple way of handling goal conflicts. A more interesting method would be to give more credence to requests from certain individuals, or requests of certain types. For example, if an agent gets a request from its owner that conflicts with a previous request from someone else, it should drop the previous request and adopt its owner's request instead. We reserve a more sophisticated handling of conflicting requests for future work.

Here we assume that request actions are always possible to execute, although executability conditions could easily be added, if desired.

Axiom 3.2

$$Poss(\text{REQUEST}(reqr, agt, \psi), s).$$

We now turn our attention to goal contraction. Suppose that the owner of an agent asks it to do ψ and later decides it no longer wants the agent to do ψ. The owner should be able to tell the agent to stop working on ψ. We use the action $\text{CANCELREQUEST}(requester, agt, \psi)$ for this purpose. This action causes agt to drop the goal that ψ. A CANCELREQUEST action can only be executed if a corresponding REQUEST action has occurred in the past.

Axiom 3.3

$$\begin{aligned}&Poss(\text{CANCELREQUEST}(requester, agt, \psi), s) \equiv\\ &do(\text{REQUEST}(requester, agt, \psi), s') \preceq s\end{aligned}$$

We handle CANCELREQUEST actions by determining what the W relation would have been if the corresponding REQUEST action had never happened. Suppose a $\text{CANCELREQUEST}(Requester, Agt, \psi)$ action occurs in situation S. We restore the W relation to the way it was before the corresponding REQUEST occurred. Then, starting just after the REQUEST, we look at all the situations $do(A^*, S^*)$ in the history of S and remove from W any situation S' that satisfies $W^-(Agt, A, S', S^*)$; such situations S' should be removed

[8]Note that according to the definition of **Goal**, if the agent has a goal that implies $\neg\psi$ then it also has the goal that $\neg\psi$, i.e., if $\textbf{Goal}(Agt, \psi', S)$ holds and $\forall s, s'.\psi'[s, s'] \supset \neg\psi[s, s']$ holds, then $\textbf{Goal}(Agt, \neg\psi, S)$ also holds. In other words, if the agent has a conflicting goal that implies $\neg\psi$, then ψ will not be adopted as a goal.

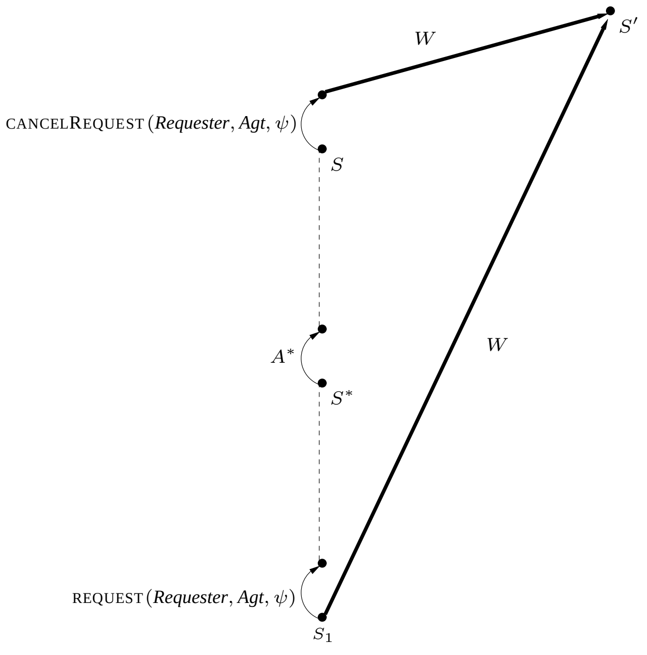
W
S'
CANCELREQUEST (Requester, Agt, ψ)
S
A*
S*
W
REQUEST (Requester, Agt, ψ)
S1

Figure 1: An example of goal contraction.

to reflect the adoption of a subsequent request. We first define the predicate $\text{CANCELS}(a, a')$, which says that action a cancels the action a'. In our case, the only cancelling action is CANCELREQUEST, and it only cancels the corresponding REQUEST:

$$\begin{aligned}&\text{CANCELS}(a, a') \overset{\text{def}}{=} \exists requester, \psi.\\ &\quad a = \text{CANCELREQUEST}(requester, agt, \psi) \wedge\\ &\quad a' = \text{REQUEST}(requester, agt, \psi)\end{aligned}$$

We now define $W^+(agt, a, s', s)$ which states the conditions under which s' is added to W^+ after a is executed in s:

$$\begin{aligned}&W^+(agt, a, s', s) \overset{\text{def}}{=}\\ &\quad(\exists s_i.W(agt, s', s_i) \wedge\\ &\qquad(\exists a_1.do(a_1, s_i) \preceq s \wedge \text{CANCELS}(agt, a, a_1) \wedge\\ &\qquad\quad\forall a^*, s^*.do(a_1, s_i) \prec do(a^*, s^*) \preceq s \supset\\ &\qquad\qquad\neg W^-(agt, a^*, s', s^*)))\end{aligned}$$

To help explain this definition, we will refer to Fig. 1, where a segment of a situation forest is shown. The situation S' is not W-related to S. However, S' was W-related to S_1, which is in the history of S. The next action after S_1 in the history of S was $\text{REQUEST}(Requester, Agt, \psi)$. We suppose S' was dropped from W after the REQUEST. If none of the actions between $do(\text{REQUEST}(Requester, Agt, \psi), S_1)$ and S also cause S' to be dropped from W, then S' is returned to W after the $\text{CANCELREQUEST}(Requester, Agt, \psi)$ action is executed in S. In other words, $W^+(Agt, \text{CANCELREQUEST}(Requester, Agt, \psi), S', S)$ holds because the following hold:

1. $W(Agt, S', S_1)$,

2. $do(\textsc{request}(Requester, Agt, \psi), S_1) \preceq S$,
3. $\textsc{cancelRequest}(Requester, Agt, \psi)$ cancels $\textsc{request}(Requester, Agt, \psi))$, and
4. for every situation $do(A^*, S^*)$ between $do(\textsc{request}(Requester, Agt, \psi), S_1)$ and S, $\neg W^-(Agt, A^*, S', S^*)$ holds.

Note that for our definition of W^+ to work properly, we must assume that there is only one REQUEST action in the history that is cancelled by each CANCELREQUEST, so we make that assumption here. We will relax this assumption in future work.

4 Properties

We now show that our formalization of goals has desirable properties. Let Σ consist of the foundational axioms, the encoding axioms, unique names actions for actions, and the axioms from Sections 2 and 3.

4.1 Goal Expansion

First, we show that our theory supports goal expansion. If an agent does not have $\neg\psi$ as a goal, then it will adopt ψ as a goal, if it receives a request for ψ.

Theorem 4.1

$$\begin{aligned}\Sigma \models \forall agt, \psi, requester, s. \neg\mathbf{Goal}(agt, \neg\psi, s) \supset \\ \mathbf{Goal}(agt, \psi, do(\textsc{request}(requester, agt, \psi), s))\end{aligned}$$

We also show the conditions under which an agent's only goals are expanded. If an agent has the only goal that ψ, ψ' is requested of the agent, the agent knows that the request does not affect the truth value of ψ, and ψ' does not conflict with the agent's goals, then the agent's only goals are expanded to include ψ'.

Theorem 4.2

$$\begin{aligned}&\Sigma \models \forall a, agt, \psi, \psi', requester, s.\\ &\quad \mathbf{OGoal}(agt, \psi, s) \land a = \textsc{request}(requester, agt, \psi') \land\\ &\quad \mathbf{Know}(agt, [\forall then.Poss(a, Now) \supset\\ &\qquad\qquad (\psi[Now, then] \equiv \psi[do(a, Now), then])], s) \land\\ &\quad \neg\mathbf{Goal}(agt, \neg\psi', s) \supset\\ &\qquad \mathbf{OGoal}(agt, \psi \land \psi', do(a, s)).\end{aligned}$$

4.2 Goal Persistence

Next, we examine the persistence of goals. A goal ψ persists over an action a, if a is not a CANCELREQUEST action, and the agent knows that a change the value of ψ, if it holds currently.

Theorem 4.3

$$\begin{aligned}&\Sigma \models \forall a, agt, \psi, s.\\ &\quad \mathbf{Goal}(agt, \psi, s) \land\\ &\quad (\forall reqr, \psi'.a \neq \textsc{cancelRequest}(reqr, agt, \psi')) \land\\ &\quad \mathbf{Know}(agt, [\forall then.Poss(a, Now) \land \psi[Now, then] \supset\\ &\qquad\qquad \psi[do(a, Now), then]], s) \supset\\ &\qquad \mathbf{Goal}(agt, \psi, do(a, s)).\end{aligned}$$

We have a similar result for "only goals".

Theorem 4.4

$$\begin{aligned}&\Sigma \models \forall a, agt, \psi, s.\\ &\quad \mathbf{OGoal}(agt, \psi, s) \land\\ &\quad [\forall requester, \psi'.\\ &\qquad (a \neq \textsc{request}(requester, agt, \psi') \lor\\ &\qquad \mathbf{Goal}(agt, \neg\psi', s)) \land\\ &\qquad a \neq \textsc{cancelRequest}(requester, agt, \psi')] \land\\ &\quad \mathbf{Know}(agt, [\forall then.Poss(a, Now) \supset\\ &\qquad (\psi[Now, then] \equiv \psi[do(a, Now), then])], s) \supset\\ &\qquad \mathbf{OGoal}(agt, \psi, do(a, s)).\end{aligned}$$

4.3 Goal Contraction

We now show a property about only goal contraction. Suppose that the agent only has the goal that ψ in s, and just after s, $requester$ requests ψ' of the agent. In some future situation s'', $requester$ cancels its request that ψ', yielding the situation s'. We assume that there was only one request of the agent for ψ' from $requester$, and that no other REQUEST or CANCELREQUEST actions occurred between s and s'. We also assume that ψ always persists (this condition can be weakened; we assumed it to simplify the proof). Then, we can show that the agent only has the goal that ψ in s. In other words, the request for ψ' was indeed cancelled, and the other "only goals" persisted.

Theorem 4.5

$$\begin{aligned}&\Sigma \models \forall agt, \psi, \psi', reqr, s, s', s''.\\ &\quad \mathbf{OGoal}(agt, \psi, s) \land\\ &\quad do(\textsc{request}(reqr, agt, \psi'), s) \preceq s' \land\\ &\quad s' = do(\textsc{cancelRequest}(reqr, agt, \psi'), s'') \land\\ &\quad [\forall s_1, s_2.do(\textsc{request}(reqr, agt, \psi'), s_1) \preceq s'' \land\\ &\qquad\quad do(\textsc{request}(reqr, agt, \psi'), s_2) \preceq s'' \supset\\ &\qquad\qquad s_1 = s_2] \land\\ &\quad [\forall s^*, a^*.\\ &\qquad do(\textsc{request}(reqr, agt, \psi'), s) \prec do(a^*, s^*) \land\\ &\qquad do(a^*, s^*) \preceq s'' \supset\\ &\qquad\quad (\neg\exists reqr', \psi''.\\ &\qquad\qquad a^* = \textsc{cancelRequest}(reqr', agt, \psi'') \lor\\ &\qquad\qquad a^* = \textsc{request}(reqr', agt, \psi''))] \land\\ &\quad [\forall a, s_1, s_2.\psi(s_1, s_2) \equiv \psi(do(a, s_1), s_2)] \supset\\ &\qquad \mathbf{OGoal}(agt, \psi, s').\end{aligned}$$

4.4 Goal Introspection

Just as agents introspect their knowledge, we want agents to be able to introspect their goals, i.e., if an agent has a goal (does not have a goal, resp.) that ψ, it should know that it has (does not have, resp.) ψ as a goal. We identify constraints that yield these properties. They are constraints on K and W.

We will need the following definitions of transitivity and Euclideanness, starting in a situation s. Note that the order of the situations in the definitions is reversed from the traditional definitions.

$$\begin{aligned}&\mathit{Ktrans}(agt,s) \overset{\text{def}}{=} \forall s_1, s_2.K(agt,s_1,s) \wedge K(agt,s_2,s_1) \supset\\ &\qquad\qquad K(agt,s_2,s).\\ &\mathit{Keuc}(agt,s) \overset{\text{def}}{=} \forall s_1, s_2.K(agt,s_1,s) \wedge K(agt,s_2,s) \supset\\ &\qquad\qquad K(agt,s_2,s_1).\end{aligned}$$

Theorem 6 of Scherl and Levesque [2003] showed that if these properties hold initially, then the successor state axiom for K guarantees that they are preserved over executable sequences of actions. We repeat this theorem here using a different notation (and we omit reference to symmetry). Note that *root*(s) is defined to be the initial situation that precedes s.

Theorem 4.6 (Scherl and Levesque [2003])

$$\begin{aligned}&\Sigma \models \forall agt, s.(\mathit{Executable}(s) \wedge \mathit{Ktrans}(agt,\mathit{root}(s)) \supset\\ &\qquad \mathit{Ktrans}(agt,s)) \wedge\\ &\qquad (\mathit{Executable}(s) \wedge \mathit{Keuc}(agt,\mathit{root}(s)) \supset\\ &\qquad \mathit{Keuc}(agt,s)),\\ &\text{where } \mathit{Executable}(s) \overset{\text{def}}{=} \forall a, s'.\mathit{do}(a,s') \preceq s \supset \mathit{Poss}(a,s').\end{aligned}$$

Similarly, we can show that reflexivity is preserved. Recall that we asserted that K was initially reflexive in Axiom 2.3. The successor state axiom for K preserves reflexivity over all executable sequences of actions.

Theorem 4.7 (Scherl and Levesque [2003])

$$\Sigma \models \forall agt, s.\mathit{Executable}(s) \supset K(agt,s,s).$$

For positive introspection of goals, we need a constraint similar to transitivity, but that involves both K and W. We call this constraint *cross-transitivity* (*CrossTrans*).

$$\begin{aligned}&\mathit{CrossTrans}(agt,s) \overset{\text{def}}{=}\\ &\quad \forall s_1, s_2, s_3.K(agt,s_1,s) \wedge K(agt,s_2,s_1) \wedge\\ &\qquad W(agt,s_3,s_1) \wedge s_2 \preceq s_3 \supset\\ &\qquad\quad W(agt,s_3,s).\end{aligned}$$

If this constraint is satisfied, and K is transitive, then the agents will have positive introspection of goals:

Theorem 4.8

$$\begin{aligned}&\Sigma \models \forall agt, s, \psi.\mathit{Ktrans}(agt,s) \wedge \mathit{CrossTrans}(agt,s) \supset\\ &\qquad (\mathbf{Goal}(agt,\psi,s) \supset\\ &\qquad\quad \mathbf{Know}(agt,\mathbf{Goal}(agt,\psi),s)).\end{aligned}$$

For negative introspection of goals, we need a constraint similar to Euclideanness, which we call *cross-Euclideanness* (*CrossEuc*):

$$\begin{aligned}&\mathit{CrossEuc}(agt,s) \overset{\text{def}}{=}\\ &\quad \forall s_1, s_2, s_3.K(agt,s_1,s) \wedge K(agt,s_2,s) \wedge\\ &\qquad W(agt,s_3,s) \wedge s_2 \preceq s_3 \supset\\ &\qquad\quad W(agt,s3,s_1).\end{aligned}$$

If this constraint is satisfied, and K is Euclidean, then the agents will have negative introspection of goals:

Theorem 4.9

$$\begin{aligned}&\Sigma \models \forall agt, s.\mathit{Keuc}(agt,s) \wedge \mathit{CrossEuc}(agt,s) \supset\\ &\qquad (\neg\mathbf{Goal}(agt,\psi,s) \supset\\ &\qquad\quad \mathbf{Know}(agt,\neg\mathbf{Goal}(agt,\psi),s)).\end{aligned}$$

We can show that *CrossTrans* and *CrossEuc* persist if they hold of the initial situations, and K is initially transitive and Euclidean.

Theorem 4.10

$$\begin{aligned}&\Sigma \models \forall agt, s.\\ &\quad \mathit{Executable}(s) \wedge\\ &\quad [\forall s'.\mathit{Init}(s') \supset\\ &\qquad (\mathit{Ktrans}(agt,s') \wedge \mathit{Keuc}(agt,s') \wedge\\ &\qquad\ \mathit{CrossTrans}(agt,s') \wedge \mathit{CrossEuc}(agt,s'))] \supset\\ &\quad \mathit{CrossTrans}(agt,s) \wedge \mathit{CrossEuc}(agt,s).\end{aligned}$$

It follows that positive and negative goal introspection persist, if the associated constraints hold initially.

5 Meeting Scheduler Example

To illustrate goal change, we now summarize the meeting scheduler example from Shapiro [2005] that is based on the one given in Shapiro *et al.* [1998], but was modified to include goal contraction using CANCELREQUEST.

We specify the behavior of agents with the notation of the programming language ConGolog [De Giacomo *et al.*, 2000], the concurrent version of Golog [Levesque *et al.*, 1997]. While versions of both Golog and ConGolog have been implemented, we are mainly interested here in the potential for using ConGolog as a specification language. The language contains the following constructs:[9]

a,	primitive action
$\phi?$,	wait for a condition
$\delta_1;\delta_2$,	sequence
$\delta_1 \mid \delta_2$,	nondeterministic choice of programs
$\pi x.\delta$,	nondeterministic choice of arguments
δ^*,	nondeterministic iteration
if ϕ **then** δ_1 $\{$**else** $\delta_2\}$ **endIf**,	conditional
for $x \in L$ **do** δ **endFor**,	for loop
while ϕ **do** δ **endWhile**,	while loop
$\delta_1 \parallel \delta_2$,	concurrency with equal priority
$\delta_1 \rangle\!\rangle\ \delta_2$,	concurrency with δ_1 at a higher priority
$\langle\ \vec{x} : \phi \rightarrow \delta\ \rangle$,	interrupt

[9]De Giacomo et. al. [De Giacomo *et al.*, 2000] allow recursive procedures in the language. To simplify matters, we omit them here. We only allow non-recursive procedures and treat them as definitions.

In the above, a denotes a situation calculus action; δ, δ_1, and δ_2 stand for programs; L is a finite list; and $\vec{x}$ is a sequence of variables. These constructs are mostly self-explanatory. Intuitively, the interrupts work as follows. Whenever $\exists\vec{x}.\phi$ becomes true, δ is executed with bindings of $\vec{x}$ that satisfy ϕ; once δ has finished executing, the interrupt can trigger again.

The semantics of ConGolog programs are defined by De Giacomo et. al. [De Giacomo *et al.*, 2000]. Their semantics is based on "single steps" of computation, or *transitions*. A step here is either a primitive action or testing whether a condition holds in the current situation. They introduce two special predicates, *Final* and *Trans*, where $\mathit{Final}(\delta, s)$ denotes that program δ may legally terminate in situation s, and where $\mathit{Trans}(\delta, s, \delta', s')$ means that program δ in situation s may legally execute one step, ending in situation s' with program δ' remaining. They then define $\mathit{Do}(\delta, s, s')$ to mean that s' is a terminating situation of program δ starting in situation s:

$$\mathit{Do}(\delta, s, s') \overset{\text{def}}{=} \exists\delta'.\mathit{Trans}^*(\delta, s, \delta', s') \wedge \mathit{Final}(\delta', s'),$$

where Trans^* is the reflexive, transitive closure of *Trans*.

In other words, $\mathit{Do}(\delta, s, s')$ holds if it is possible to repeatedly single-step the program δ, obtaining a program δ' and a situation s' such that δ' can legally terminate in s'. The semantics does not handle the for-loop construct, since De Giacomo et. al. did not have this construct. However, it is straightforward to extend their semantics to handle for-loops.

In the meeting scheduler example [Shapiro *et al.*, 1998; Shapiro and Lespérance, 2001; Shapiro, 2005], there are meeting organizer agents, which are trying to schedule meetings with personal agents, which manage the schedules of their (human) owners. To schedule a meeting, an organizer agent requests of each of the personal agents of the participants in the meeting to adopt the goal that its owner attend the meeting during a given time period. If a personal agent does not have any goals that conflict with its owner attending the meeting (i.e., it has not previously scheduled a conflicting meeting), it adopts the goal that its owner attend this meeting and informs the meeting organizer that it has adopted this goal, i.e., that it accepts the meeting request. Otherwise, the personal agent informs the meeting organizer that it has not adopted the goal that its owner attend the meeting, i.e., that it declines the meeting request.

The procedure that defines the behaviour of meeting organizer agents ($\mathrm{organizeMeeting}$) is given in Fig. 2.[10] Their task is to organize a meeting for a given period and set of participants on behalf of the chair of the meeting. They do this by asking the personal agent of each participant to have their owner meet during the given period. The participants' personal agent will then reply whether they have declined the meeting. The behaviour of the personal agents is specified by the MANAGESCHEDULE procedure which is not shown here due to space constraints. In that procedure, a personal agent declines the meeting if it has already scheduled a conflicting meeting. Otherwise, it adopts the goal that its owner be at the requested meeting. The meeting organizer waits until it knows whether one of the participants' agent declined the meeting. This will happen when either someone has declined the meeting or everyone has agreed to it. If the organizer finds out that someone declined the meeting, it cancels its request for a meeting with the personal agents of all the participants in the meeting. The personal agents that agreed to the meeting will then drop the goal that their owners be at the requested meeting.

A complete meeting scheduler system is defined by composing instances of the personal agents and the meeting organizer agents in parallel, thereby modelling the behavior of several agents acting independently. We also need to compose the nondeterministic iteration of a *tick* action concurrently (at a lower priority) to allow time to pass when the agents are not acting. Here is an example of such a system:

$$\begin{aligned}
&[\mathrm{MANAGESCHEDULE}(\mathrm{PA}_1, \mathrm{USER}_1) \parallel \\
&\mathrm{MANAGESCHEDULE}(\mathrm{PA}_2, \mathrm{USER}_2) \parallel \\
&\mathrm{MANAGESCHEDULE}(\mathrm{PA}_3, \mathrm{USER}_3) \parallel \\
&\mathrm{ORGANIZEMEETING}(\mathrm{OA}_1, \mathrm{USER}_1, \{\mathrm{USER}_1, \mathrm{USER}_3\}, \\
&\qquad\qquad \text{12:00–2:00}) \parallel \\
&\mathrm{ORGANIZEMEETING}(\mathrm{OA}_2, \mathrm{USER}_2, \{\mathrm{USER}_2, \mathrm{USER}_3\}, \\
&\qquad\qquad \text{1:30–2:45})] \rangle\!\rangle\ \mathrm{TICK}^*
\end{aligned}$$

6 Conclusions and Future Work

In this paper, we defined the goals and "only goals" of an agent, and examined properties of these definitions: expansion, contraction, and persistence. We identified the restrictions on K and W that give us introspection of goals, and showed that these restrictions persist, if they hold initially. Note that our approach is quite different and simpler than most existing approaches to belief change. These require a specification of the plausibilities of alternate situations. Our approach avoids this and instead relies on the history to undo previous goal expansions. However, we don't handle arbitrary goal contractions, only cancellations of previous requests. Note there are some related approaches in the belief change literature [Booth *et al.*, 2005; Roorda *et al.*, 2003]. This raises the question of using belief change frameworks to handle goal change, or applying our goal change approach to belief change. It would be interesting to investigate whether there are essential differences between belief revision and goal revision that warrant different solutions, or whether the same solutions can be applied to both problems. Also, it would be interesting to identify postulates for goal change and examine how they differ from belief change postulates [Gärdenfors, 1988]. Our handling of the cancelling of requests needs further work and analysis. First of all, it does not properly handle nested requests for the same goal. For the successor state axiom to work properly for goal contraction, we must assume that there is only one REQUEST action in the history that is cancelled by each CANCELREQUEST. This would not be too difficult to remedy, but it leads us to another difficulty. The definition of W^+ is already quite complex, which makes proving properties about it somewhat

[10]Note that in the definition of this procedure, $\mathrm{PAG}(p)$ is a function that maps an owner to its personal agent, MEMBER(p,$parts$) holds if p is a member of the list $parts$, and $\mathbf{KWhether}(agt, \phi, s) \overset{\text{def}}{=} \mathbf{Know}(agt, \phi, s) \vee \mathbf{Know}(agt, \neg\phi, s)$.

```
$\textsc{informWhether}(agt, agt', \phi) = [\textbf{Know}(agt, \phi)?; \textsc{inform}(agt, agt', \phi)]\ |$
  $[\textbf{Know}(agt, \neg\phi)?; \textsc{inform}(agt, agt', \neg\phi)]\ |\ [\neg\textbf{KWhether}(agt, \phi)?; \textsc{inform}(agt, agt', \neg\textbf{KWhether}(agt, \phi))]$

$\mathrm{SomeoneDeclined}(per, chair, parts, s) =$
  $\exists p.\textsc{member}(p, parts) \wedge \neg\textbf{Goal}(\textsc{pag}(p), \textbf{During}(per, \textsc{AtMeeting}(p, chair)), s)$

$\mathrm{organizeMeeting}(oa, chair, parts, per) =$
  **for** $p \in parts$ **do**
    $\textsc{request}(oa, \textsc{pag}(p), \textbf{During}(per, \textsc{AtMeeting}(p, chair)))$ **endFor**;
  $\textbf{KWhether}(oa, \mathrm{SomeoneDeclined}(per, chair, parts))?$;
  **if** $\textbf{Know}(oa, \mathrm{SomeoneDeclined}(per, chair, parts))$ **then**
    **for** $p \in parts$ **do**
      $\mathrm{cancelRequest}(oa, \textsc{pag}(p), \textbf{During}(per, \textsc{AtMeeting}(p, chair)))$ **endFor endIf**;
  $\textsc{informWhether}(oa, chair, \mathrm{SomeoneDeclined}(per, chair, parts))$
```

Figure 2: Procedure run by the meeting organizer agents.

awkward. Since the definition refers to a sequence of situations, the proofs will often involve inductions. We would like to investigate the possibility of simplifying the definition.

References

[Booth *et al.*, 2005] R. Booth, S. Chopra, A. Ghose, and T. Meyer. Belief liberation (and retraction). *Studia Logica*, 79:47–72, 2005.

[Cohen and Levesque, 1990] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[De Giacomo *et al.*, 2000] G. De Giacomo, Y. Lespérance, and H. J. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121(1–2):109–169, 2000.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux*. The MIT Press: Cambridge, MA, 1988.

[Konolige and Pollack, 1993] K. Konolige and M. E. Pollack. A representationalist theory of intention. In *Proc. IJCAI-93*, pages 390–395, 1993.

[Levesque *et al.*, 1997] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[Levesque *et al.*, 1998] H. J. Levesque, F. Pirri, and R. Reiter. Foundations for a calculus of situations. *Electronic Transactions of AI (ETAI)*, 2(3–4):159–178, 1998.

[McCarthy and Hayes, 1969] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*. Edinburgh University Press, 1969.

[Moore, 1985] R. C. Moore. A formal theory of knowledge and action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Common Sense World*, pages 319–358. Ablex Publishing, Norwood, NJ, 1985.

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

[Roorda *et al.*, 2003] J.-W. Roorda, W. van der Hoek, and J.-J. Meyer. Iterated belief change in multi-agent systems. *Logic Journal of the IGPL*, 11(2):223–246, 2003.

[Scherl and Levesque, 2003] R. B. Scherl and H. J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.

[Shapiro and Lespérance, 2001] S. Shapiro and Y. Lespérance. Modeling multiagent systems with the cognitive agents specification language — a feature interaction resolution application. In C. Castelfranchi and Y. Lespérance, editors, *Proc. ATAL-2000*, pages 244–259. Springer-Verlag, Berlin, 2001.

[Shapiro *et al.*, 1998] S. Shapiro, Y. Lespérance, and H. J. Levesque. Specifying communicative multi-agent systems. In W. Wobcke, M. Pagnucco, and C. Zhang, editors, *Agents and Multi-Agent Systems — Formalisms, Methodologies, and Applications*, volume 1441 of *LNAI*, pages 1–14. Springer-Verlag, Berlin, 1998.

[Shapiro *et al.*, 2000] S. Shapiro, M. Pagnucco, Y. Lespérance, and H. J. Levesque. Iterated belief change in the situation calculus. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proc. KR-2000*, pages 527–538, 2000.

[Shapiro *et al.*, 2002] S. Shapiro, Y. Lespérance, and H. J. Levesque. The cognitive agents specification language and verification environment for multiagent systems. In C. Castelfranchi and W. L. Johnson, editors, *Proc. AAMAS-02*, pages 19–26. ACM Press, 2002.

[Shapiro, 2005] S. Shapiro. PhD thesis. To appear., 2005.

[Spohn, 1988] W. Spohn. Ordinal conditional functions: A dynamic theory of epistemic states. In W. Harper and B. Skyrms, editors, *Causation in Decision, Belief Change, and Statistics*, pages 105–134. Kluwer Academic Publishers, 1988.